Water Wave Optimization for Artificial Neural Network Parameter and Structure Optimization

Xiao-Han Zhou[1], Zhi-Ge Xu[1], Min-Xia Zhang[1], and Yu-Jun Zheng[1,2](✉)

[1] College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310023, China

[2] Institute of Service Engineering, Hangzhou Normal University, Hangzhou 311121, China

seanzxhan@gmail.com, xuzhige@foxmail.com, zmx@zjut.edu.cn, yujun.zheng@computer.org

Abstract. Artificial neural networks (ANNs) have powerful function approximation and pattern classification capabilities, but their performance is greatly affected by structural design and parameter selection. Traditional training methods, have drawbacks including long training time, over-fitting, premature convergence, etc. Evolutionary optimization algorithms have provided an effective tool for ANN parameter optimization, but simultaneously optimizing ANN structure and parameters remains a difficult problem. This paper adapts a relatively new evolutionary algorithm, water wave optimization (WWO), for both structure design and parameter selection for ANNs. The algorithm uses a variable-dimensional solution representation, and designs new propagation, refraction, and breaking operators to effectively evolve solutions towards the optimum or near-optima. Computational experiments show that the WWO algorithm exhibits significant performance advantages over other popular evolutionary algorithms including genetic algorithm, particle swarm optimization, and biogeography-based optimization, for ANN structure and parameter optimization.

Keywords: Artificial neural networks (ANNs), evolutionary neural networks, parameter selection, structural design, water wave optimization (WWO)

1 Introduction

Since the original work of McCulloch and Pitts [7] in the 1940s, artificial neural networks (ANNs) have become an active research field in artificial intelligence. Nevertheless, the performance of ANNs is seriously affected by the selection of their structures and parameters. Traditional training methods, such as the back-propagation (BP) algorithm [3], have drawbacks such as long training time, over-fitting, and premature convergence. To overcome these disadvantages, a number of heuristic optimization algorithms, in particular bio-inspired evolutionary algorithms, have been applied to and shown good performance on ANN training [9]. Essentially, the problem of selecting parameters (including neuron biases and

connection weights) for an ANN can be regarded as a high-dimensional global optimization problem [13], for which metaheuristic algorithms including genetic algorithms (GA) [12], evolutionary programming (EP) [11], particle swarm optimization (PSO) [8, 14], biogeography-based optimization (BBO) [15], etc., can efficiently explore the search space.

Nevertheless, simultaneously optimizing the structure and parameters of an ANN is much more complex. Lam et al. [5] use an improved GA to tune ANN structure and parameters, where number of hidden nodes starts from a small number and will continue to increase if the fitness is not acceptable. However, such a tuning method is very time-consuming. Kiranyaz et al. [4] propose a multi-dimensional PSO algorithm that removes the necessity of fixing the dimension in advance and thus enables encoding both network structure and parameters into particles. Das et al. [2] propose another PSO algorithm for ANN training, where the number of layers and neurons are also regarded as parameters encoded into particles, but this also makes the search space extremely huge and thus degrades the algorithm performance. Salama and Abdelbar [10] propose an ant colony optimization (ACO) algorithm to learn ANN structure, where the number of hidden neurons is determined by pruning the maximally connected network structure, but it does not simultaneously optimize the structure and parameters. In general, simultaneously optimizing ANN structure and parameters remains a quite difficult optimization problem.

In this paper, we propose a new water wave optimization (WWO) algorithm for ANN structure and parameters optimization. The algorithm uses a variable-dimensional solution representation to encode both structure configurations and neuron/connection parameters, and efficiently explore the huge solution space based on adapted propagation, refraction, and breaking operators. Computational experiments show that the WWO algorithm exhibits competitive performance on ANN parameter & structure optimization compared to other popular metaheuristics.

In the rest of this paper, Section 2 describes the problems of ANN structural design and parameter selection. Section 3 proposes the WWO algorithm. Section 4 presents the computational experiments, and finally Section 5 concludes.

2 ANN Parameter & Structure Optimization

2.1 The Problem of ANN Parameter Optimization

In this paper we focus on the most widely-used feedforward neural networks, where each layer contains a set of artificial neurons that can only be connected to the neurons of the next layer. However, it is not difficult to extend our approach for other types of ANNs. In a feedforward ANN, each neuron in the input layer directly accepts an input, while each neuron j in the hidden or output layer accepts a set of inputs x_{ij} from the neurons of the previous layer and produces an output y_j as follows:

$$y_j = \phi(\sum_{i=1}^{m} w_{ij} x_{ij} - \theta_i) \tag{1}$$

where m is the number of inputs, w_{ij} is the connection weight of ith input, θ_j is the threshold of the neuron, and ϕ is the activation function which typically uses the Sigmoid function or the hyperbolic tangent function.

Once the structure of the network has been fixed, we need to tune the parameters including all neuron biases θ_i and all connection weights w_{ij}, such that the actual output of the ANN on the training set is as close as possible to the expected output. Suppose that the training set has N samples, the input-output pair of each sample is $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, the output of the ANN for $\boldsymbol{x}_i$ is $\boldsymbol{o}_i$, the ANN training is to minimize the root mean square error (RMSE) between the actually outputs and the expected outputs:

$$\min \mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{y}_i - \boldsymbol{o}_i\|^2} \tag{2}$$

Thus the ANN training problem can be regarded as a high-dimensional optimization problem. Typically, the values of biases and weights can be limited in the range of $[0, 1]$ or $[-1, 1]$. For example, for a typical three-layer feedforward ANN, let n_1, n_2, and n_3 be the number of neurons in the input layer, the hidden layer, and the output layer, respectively, the problem dimension is $(n_1 n_2 + n_2 n_3 + n_1 + n_2 + n_3)$.

2.2 The Problem of ANN Structure Optimization

The above problem assumes that the ANN structure has been fixed, and only neuron biases and connection weights are to be optimized. However, ANN performance also heavily depends on its structure. When the input-output scheme of an ANN is known, we need to determine the number of neurons in each hidden layer. For example, for a three-layer ANN, if the number n_2 of neurons in the hidden layer is too small, the computing power of the ANN will be limited, and the training will be easily trapped in local optima; on the contrary, if n_2 is too large, the training time will be prolonged, and the training results are more likely to be over-fitting. However, in traditional approaches the number of neurons is determined mainly based on experiences, which often leads to inappropriate structural design.

Therefore, it is reasonable to simultaneously optimize the structure and parameters of ANN. For a three-layer ANN, the number n_2 of neurons in the hidden layer should be an additional decision variable to be optimized. Empirically, the value range of n_2 can be set as follows:

$$\log_2 n_1 \le n_2 \le \sqrt{n_1 + n_3} + 10 \tag{3}$$

Besides the minimization of RMSE on the training set, we also expect that n_2 can be as small as possible in order to simplify the network structure, and thus we can set the objective function as follows:

$$\min f = w \cdot \mathrm{RMSE} + (1 - w)\frac{n_2}{n_1 + n_3 + \sqrt{n_1 + n_3} + 10} \tag{4}$$

where w represents the weight of RMSE in the objective function, and its value is typically between 0.6 and 0.9.

By analogy, for a K-layer ANN, we need to determine $(K-2)$ additional decision variables, i.e., $n_2, n_3, \ldots, n_{K-1}$the objective function as follows:

$$\min f = w \cdot \mathrm{RMSE} + (1-w)\frac{n_2 + n_3 + \cdots + n_{K-1}}{n_2^{\mathrm{U}} + n_3^{\mathrm{U}} + \cdots + n_{K-1}^{\mathrm{U}}} \tag{5}$$

where n_k^{U} denotes the upper limit of the number of neurons in the k-th layer.

Although structural design itself adds only a small number of decision variables, it makes the number of other bias and weight parameters variable, and thus makes the integrated ANN structure and parameter optimization problem become a variable-dimensional optimization problem that is much more difficult to solve.

3 A New WWO Algorithm for ANN Parameter & Structure Optimization

In this paper we propose a new WWO algorithm for ANN optimization. WWO [17] is a relatively new metaheuristic that takes inspiration from shallow water wave models for optimization, where each solution X is analogous to a wave. The higher (lower) the energy or fitness of the wave, the smaller (larger) the wavelength λ_X, and the smaller (larger) the range it explores, as shown in Fig. 1. In this way, the algorithm can achieve a good dynamic balance between diversification and intensification.

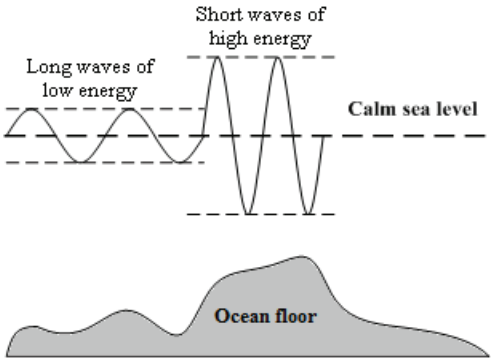

Fig. 1: Illustration of wave propagation in shallow water.

3.1 WWO for ANN Parameter Optimization

The original WWO uses three operators, named propagation, refraction, and breaking, for searching in high-dimensional continuous solution space, and thus can directly apply to the ANN parameter optimization problem.

Propagation of a solution X is done by shifting each dimension d of X as follows (where function *rand* produces a random number uniformly distributed in the given range, and $L(d)$ is the length of the dth dimension of the search space):

$$X'(d) = X(d) + \lambda_X \cdot rand(-1, 1) \cdot L(d) \tag{6}$$

All wavelengths are initially set to 0.5, and then updated after each generation as follows:

$$\lambda_X = \lambda_X \cdot \alpha^{-(f(X)-f_{\min}+\epsilon)/(f_{\max}-f_{\min}+\epsilon)} \tag{7}$$

where $f_{\max}$ and $f_{\min}$ are respectively the maximum and minimum fitness values among the population, α is the wavelength reduction coefficient set to 1.0026, and ϵ is a very small number to avoid division-by-zero.

Remark 1. f in Eq. (7) denotes the fitness function. For a problem minimizing an objective function as Eq. (4) or (5), the numerator of the exponential term in Eq. (7) can be replaced by $-(f_{\max} - f(X) + \epsilon)$.

Refraction replaces a wave X losing energy (i.e., having not been improved after several generations) with a new wave generated at a random position between the old wave and the best known solution X^*:

$$X'(d) = \mathcal{N}\big(\frac{X^*(d) + X(d)}{2}, \frac{|X^*(d) - X(d)|}{2}\big) \tag{8}$$

where $\mathcal{N}(\mu, \sigma)$ generates a Gaussian random number with mean μ and standard deviation σ.

Breaking breaks any newly found best wave X^* into several solitary waves, each of which moves a small distance from X^* at a random direction:

$$X'(d) = X^*(d) + \mathcal{N}(0, 1) \cdot \beta L(d) \tag{9}$$

where β is the breaking coefficient. The best solitary wave, if better than X^*, will replace X^* in the population.

Alg. 1 presents the framework of WWO. If the number of neuron biases and connection weights is n, each solution to the ANN parameter optimization problem is an n-dimensional real-valued vector.

3.2 WWO for ANN Structure & Parameter Optimization

When simultaneously optimizing ANN structure and parameters, we use a variable-dimensional solution representation. For a K-layer ANN, the number of hidden layers is $(K-2)$, and each solution to the ANN structure & parameter optimization problem consists of two parts:

Algorithm 1: The WWO algorithm.

```
Randomly initialize a population P of solutions;
while the stop criterion is not satisfied do
    foreach X ∈ P do
        Propagate X to a new X' based on Eq. (6);
        if f(X') < f(X) then
            X ← X';
            if f(X) < f(X*) then
                X* ← X;
                Break X* based on Eq. (9);
        else
            if X has not been updated for ĝ consecutive generations then
                Refract X to a new X' based on Eq. (8);
    Update the wavelengths based on Eq. (7);
return X*.
```

- The first part has $(K-2)$ integer components $\{n_2, n_3, \ldots, n_{K-1}\}$, each representing the number of neurons in the corresponding hidden layer.
- The second part has $\left(\sum_{i=1}^{K} n_i\right) + \left(\sum_{i=1}^{K-1} n_i n_{i+1}\right)$ real-valued components, each representing a bias or a connection weight.

When propagating a variable-dimensional solution X, we first modify the first $(K-2)$ components according to the standard propagation operation (6) and round the results into the nearest integers. For each integer component n_k, its value change can be divided into the following three cases:

Case 1. The updated $n'_k = n_k$, which will not affect other real-valued components, and these components are also updated according to the standard propagation operation (6).

Case 2. $n'_k < n_k$, which indicates that the number of neurons decreases. Those real-valued components related to the remaining neurons are updated according to Eq. (6), and the other real-valued components related to the removed neurons are also removed.

Case 3. $n'_k > n_k$, which indicates that the number of neurons increases. Those real-valued components related to the original neurons are updated according to Eq. (6), and those new real-valued components related to the new neurons are randomly generated within the predefined range.

When refracting a stationary solution X based on the best known solution X^*, for each dimension k of the first $(K-2)$ integer components we also have three cases:

Case 1. $X(k) = X^*(k)$, for which the standard refraction operation (8) is directly applied.

Case 2. $X(k) > X^*(k)$, for which we set the refracted $X'(k)$ to a Gaussian random value between $[X^*(k), X(k)]$. For each corresponding real-valued dimension d in both X and X^*, the refracted values is set according to Eq. (8); for each corresponding real-valued dimension d in X but not in X^*, the refracted values is set to a Gaussian random value with mean $X(d)$ and standard deviation 0.5.

Case 3. $X(k) < X^*(k)$, for which we set the refracted $X'(k)$ be a Gaussian random value within $[X(k), X^*(k)]$. For each corresponding real-valued dimension d in both X and X^*, the refracted values is set according to Eq. (8); for each corresponding real-valued dimension d in X^* but not in X, the refracted value is set to a Gaussian random value with mean $X^*(d)$ and standard deviation 0.5.

When breaking a newly found best solution X^*, if the selected dimension d is a real-valued dimension, the standard breaking operation (9) is directly applied. If the selected dimension d is among the first $(K-2)$ integer dimensions, the breaking is done by one of the following two ways:

Case 1. $n_k = n_k - 1$, i.e., a neuron is removed from the k-th layer, and those real-valued dimensions related to the removed neuron are also removed.

Case 2. $n_k = n_k + 1$, i.e., a new neuron is added to the k-th layer, and those new real-valued components related to the new neuron are first randomly generated and then updated by applying the back-propagation algorithm.

By adapting the three operators of WWO, the algorithm can effectively solve the ANN structure & parameter optimization problem.

4 Computation Experiments

We respectively test the algorithm's performance on ANN parameter optimization, one-hidden-layer ANN structure & parameter optimization, and two hidden-layer ANN structure & parameter optimization. For ANN parameter optimization, the training task is to classify three categories of wines according to 13 attributes including alcohol content, acidity, color, etc. Accordingly, the input dimension is 13 and the output dimension is one. The number of neurons in the hidden layer is empirically set to $(13+1)/2 = 7$. The test set is the WINE dataset from the UCI Repository [1], which contains 178 samples. The experiment is divided into three groups, where the ratio of training set size to test set size is set to 2:3, 3:2, and 4:1, respectively. The proposed algorithm is compared with the traditional BP algorithm and four evolutionary learning algorithms, including GA [6], PSO [14], BBO [15], and EBO [18], and their performance is evaluated in terms of the average classification accuracy, i.e., the percentage of the number of samples that are correctly classified. The BP algorithm is run once, while each metaheuristic algorithm is run for 20 times.

Table 1 presents the classification results of the algorithms for ANN parameter optimization, where the maximum/mean classification accuracies among the

algorithms are shown in bold. As we can see from the results, the performance of GA is not always better than the traditional BP algorithm: on the first group, both the maximum and the mean accuracies of GA are smaller than BP; on the second and the third groups, the maximum accuracies of GA are larger than BP, but the mean accuracies of GA are smaller. Thus, the adoption of GA for ANN parameter optimization achieves little performance improvement, mainly because the crossover and mutation operations of GA easily lead to premature convergence. However, the other four metaheuristic algorithms show significant performance improvement over BP. On the first group, both EBO and WWO obtain the best maximum accuracy, and EBO obtains the best mean accuracy. On the second and the third groups, both the maximum and the mean accuracies of WWO are the best among all algorithms. In particular, on the third group, the ANN optimized by WWO is able to correctly classify all the 36 test samples. The results demonstrate that, even only for optimizing ANN parameters, the proposed WWO shows competitive performance on the test problem.

Table 1: Classification accuracies of three-layer ANNs trained by the algorithms for parameter optimization.

Group	Metrics	BP	GA	PSO	BBO	EBO	WWO
2:3	Max	84.11	83.18	92.52	88.79	**96.26**	**96.26**
	Min	84.11	79.44	90.65	82.24	94.39	92.52
	Mean	84.11	81.33	91.67	85.2	**95.13**	94.96
	Std	–	1.88	0.9	2.12	1.02	1.49
3:2	Max	84.27	84.87	93.28	89.08	**97.48**	**97.48**
	Min	84.27	80.67	91.60	84.87	94.96	96.64
	Mean	84.27	83.15	92.35	87.11	96.29	**96.95**
	Std	–	2.06	0.89	2.33	1.15	0.89
4:1	Max	77.78	83.33	94.44	88.89	97.22	**100.00**
	Min	77.78	75.00	94.44	80.56	97.22	97.22
	Mean	77.78	80.78	94.44	82.9	97.22	**98.05**
	Std	–	3.2	0	3.16	0	1.27

The test of one-hidden-layer ANN structure & parameter optimization is also conducted on the WINE dataset and uses three groups as described above, except that the number of neurons in the hidden layer is also optimized by the algorithms. The proposed algorithm is compared with GA [6], PSO [2], and BBO and EBO which also encode the number of neurons as a solution component [19]. Table 2 presents the classification results of the algorithms for ANN structure & parameter optimization. The results show that EBO and WWO also achieve the much better classification accuracies than the other three algorithms. Compared to Table 1, we can see that for GA and BBO, the classification accuracies obtained by optimizing both network structure and parameters are even worse than that are obtained by only optimizing parameters. This is because the inclusion of the structure parameter greatly enlarges the solution space, and GA and

BBO cannot efficiently explore such a large solution space. The other three algorithms show much better performance as they can find better network structures (if exist) than the structure set based on experience. In particular, as a major upgrade of BBO, EBO exhibits significant performance improvement over the original BBO, because the combination of global migration and local migration in EBO ha much higher exploration ability than the original clonal migration in BBO [18]. In this experimental part, both EBO and WWO can find the optimal number of neuron in the hidden layer, and they reach the same accuracies on the second and the third groups, while WWO performs slightly better than EBO on the first group.

Table 2: Classification accuracies of three-layer ANNs trained by the algorithms for structure & parameter optimization.

Group	Metrics	GA	PSO	BBO	EBO	WWO
2:3	Max	84.11	94.39	86.92	96.26	**98.13**
	Min	79.44	93.46	83.18	96.26	96.26
	Mean	81.70	93.69	84.89	96.26	**96.65**
	Std	2.01	0.71	1.61	0	0.88
3:2	Max	84.87	94.12	88.24	**97.48**	**97.48**
	Min	82.35	90.76	85.71	97.48	97.48
	Mean	83.60	92.23	86.86	**97.48**	**97.48**
	Std	1.12	1.55	1.22	0.91	0
4:1	Max	80.56	97.22	88.89	**100**	**100**
	Min	77.78	97.22	77.78	100	100
	Mean	79.32	97.22	81.38	**100**	**100**
	Std	2.05	0	5.16	0	0

The test of two hidden-layer ANN structure & parameter optimization is conducted on an airline passenger dataset from [16]. The task is to identify suspicious terrorists from normal passengers. The input dimension is around 7000, and thus using a single hidden layer is not sufficient for such a high-dimensional problem. The number of samples in the dataset is huge, and for convenience we conduct three groups of experiment which take 1000, 3000, and 6000 samples, respectively. The number of positive samples is 162 in all three groups. The five metaheuristics are the same as used in the second group. Table 3 present the experimental results. On this much more complex problem, the performance of GA is still the worst, and its accuracies are quite unstable among different runs. On the last group, the mean accuracies of GA and BBO are below 50% and cannot be acceptable in practice. The performance of EBO is much higher than GA, PSO, and BBO, but obviously worse than WWO. In summary, WWO exhibits significant performance advantages over all other four algorithms, and it uniquely obtains the best maximum and mean classification accuracies on all three groups. This demonstrates that the proposed WWO algorithm is quite efficient for complex ANN training problems.

Table 3: Classification accuracies of four-layer ANNs trained by the algorithms for structure & parameter optimization.

Group	Metrics	GA	PSO	BBO	EBO	WWO
#1	Max	70.50	73.70	69.80	79.70	**82.20**
	Min	53.20	65.80	61.10	75.10	78.00
	Mean	61.55	69.63	65.50	77.02	**79.31**
	Std	7.36	2.93	3.51	2.11	1.56
#2	Max	58.90	67.67	62.53	71.30	**74.80**
	Min	49.63	60.27	57.80	66.56	70.17
	Mean	54.21	63.39	59.91	69.03	**72.23**
	Std	4.55	3.02	1.96	2.76	2.60
#3	Max	47.53	56.57	50.13	63.20	**65.10**
	Min	39.50	51.23	46.60	57.93	60.60
	Mean	42.66	53.09	48.12	60.11	**62.39**
	Std	3.84	2.05	1.80	2.51	2.37

5 Conclusion

This paper proposes a new WWO algorithm for ANN structure and parameter optimization. The algorithm uses a variable-dimensional solution representation, and proposes new propagation, refraction, and breaking operators to effectively evolve variable-dimensional solutions towards the optimum or near-optima. Computational experiments show that the WWO algorithm exhibits competitive performance on ANN parameter & structure optimization compared to some other popular evolutionary algorithms. Our ongoing work is to adapt the algorithm for more complex ANN types, including recurrent neural networks, neuro-fuzzy systems, and deep neural networks.

Acknowledgements This work is supported by National Natural Science Foundation (Grant No. 61473263) and Zhejiang Provincial Natural Science Foundation (Grant No. LY14F030011) of China.

References

1. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
2. Das, G., Pattnaik, P.K., Padhy, S.K.: Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Syst. Appl. 41(7), 3491–3496 (2014)
3. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Nat. Academy Sci. 79, 2554–2558 (1982)
4. Kiranyaz, S., Ince, T., Yildirim, A., Gabbouj, M.: Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. Neural Netw. 22(10), 1448–1462 (2009)

5. Lam, H.K., Ling, S.H., Leung, F.H.F., Tam, P.K.S.: Tuning of the structure and parameters of neural network using an improved genetic algorithm. In: The 27th Annual Conference of the IEEE Industrial Electronics Society. vol. 1, pp. 25–30 (2001)
6. Leung, F.H.F., Lam, H.K., Ling, S.H., Tam, P.K.S.: Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans. Neural netw. 14, 79–88 (2003)
7. McCulloch, W.S., Pitts, W.H.: A logical calculus for the ideas immanent in nervous activity. Bull. Math. Biophys. 5, 115–133 (1943)
8. Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle swarms for feedforward neural network training. In: International Joint Conference on Neural Networks. vol. 2, pp. 1895–1899 (2002)
9. Ojha, V.K., Abraham, A., Snášel, V.: Metaheuristic design of feedforward neural networks: A review of two decades of research. Eng. Appl. Artif. Intell. 60, 97–116 (2017)
10. Salama, K.M., Abdelbar, A.M.: Learning neural network structures with ant colony algorithms. Swarm Intell. 9(4), 229–265 (2015)
11. Sarkar, M., Yegnanarayana, B.: Feedforward neural networks configuration using evolutionary programming. In: International Conference on Neural Networks. vol. 1, pp. 438–443 (1997)
12. Whitley, D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: optimizing connections and connectivity. Parall. Comput. 14(3), 347–361 (1990)
13. Yao, X.: A review of evolutionary artificial neural networks. Int. J. Intell. Syst. 8(4), 539–567 (1993)
14. Zhang, J.R., Zhang, J., Lok, T.M., Lyu, M.R.: A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training. Appl. Math. Comput. 185(2), 1026–1037 (2007)
15. Zhang, Y., Phillips, P., Wang, S., Ji, G., Yang, J., Wu, J.: Fruit classification by biogeography-based optimization and feedforward neural network. Expert Syst. 33(3), 239–253 (2016)
16. Zheng, Y.J., Sheng, W.G., Sun, X.M., Chen, S.Y.: Airline passenger profiling based on fuzzy deep machine learning. IEEE Trans. Neural Netw. Learn. Syst. 28(12), 2911–2923 (2017)
17. Zheng, Y.J.: Water wave optimization: A new nature-inspired metaheuristic. Comput. Oper. Res. 55(1), 1–11 (2015)
18. Zheng, Y.J., Ling, H.F., Xue, J.Y.: Ecogeography-based optimization: Enhancing biogeography-based optimization with ecogeographic barriers and differentiations. Comput. Oper. Res. 50, 115–127 (2014)
19. Zheng, Y., Chen, S., Zhang, M.: Biogeography-based optimization: Algorithms and Applications. Science Press (2016)